Modelo de *machine learning* para la cuantificación del cumplimiento de un plan de trabajo en un entorno empresarial complejo

Trabajo de Fin de Grado en Física

Curso 2020 – 2021

Autor:

Raúl Almuzara Diarte

Directores:

David Íñiguez Dieste
Francisco Marías Ferrer

Índice

1. Introducción

En los últimos años, el análisis de datos mediante sofisticadas técnicas computacionales se ha convertido en una actividad indispensable para la toma de decisiones y la optimización de procesos. Un adecuado tratamiento puede revelar patrones y descifrar información oculta en conjuntos de datos de cualquier tamaño. Tanto en el ámbito de la investigación como en el ámbito empresarial, términos como *Big Data* o *Machine Learning* resuenan con fuerza debido a la revolución que han supuesto en cuestiones de rendimiento, reducción de costes, ciberseguridad, sistemas de recomendación, etc.

En este trabajo, se pretende poner de manifiesto la eficacia de algunos algoritmos de inteligencia artificial para resolver un problema concreto propuesto por una empresa real. Se trata de Distromel S.A., la cual, entre otras cosas, desarrolla *software* para la gestión de residuos urbanos. Para la realización de este trabajo, se dispone de los datos que proporciona dicha empresa. El objetivo es diseñar una serie de modelos que permitan extraer información acerca del grado de bondad con el que se realiza la recogida de residuos urbanos a partir de unos datos de entrada.

En primer lugar, deberá realizarse una adecuada selección, filtrado y preparación de los datos que servirán para entrenar los modelos de *machine learning*. Después, debe hacerse un análisis cuidadoso de las variables más relevantes y su fundamento matemático para obtener información con una utilidad real. Finalmente, disponiendo de los datos de entrenamiento, se procede con el diseño de los modelos de inteligencia artificial que aprenderán de los datos proporcionados para poder realizar predicciones inteligentes. Estas predicciones se llevarán a cabo mediante algoritmos de clasificación (determinación de una categoría cualitativa) y de regresión (obtención de un valor numérico). Se presentará una descripción suficientemente autocontenida de conceptos estadísticos relevantes con los que construir y evaluar modelos de *machine learning*. En particular, una vez realizados la limpieza y el preprocesamiento del conjunto de datos, se crearán árboles de decisión, redes neuronales y técnicas basadas en estos para observar su modo de aprendizaje y los resultados que se pueden alcanzar según la capacidad de los modelos.

2. Planteamiento del problema

En la gestión de residuos urbanos, intervienen numerosas variables que producen enormes cantidades de datos (trayectorias GPS, localización de contenedores, cumplimiento de la planificación, etc.) que deben analizarse mediante el adecuado tratamiento estadístico y las técnicas de análisis pertinentes. La gestión inteligente de la recogida de residuos puede llevarse a cabo mediante algoritmos de inteligencia artificial que optimicen estos procesos y arrojen información de valor sobre la actividad de los vehículos de recogida.

El problema que nos concierne es el de **la cuantificación del grado de cumplimiento con el que se realiza una determinada orden de trabajo**. El procesamiento de las variables implicadas en la ejecución de las órdenes de trabajo no es trivial, así como tampoco lo es la definición de una métrica de bondad objetiva y que pueda obtenerse automáticamente. Para ello, se propone la evaluación de distintos algoritmos de inteligencia artificial que procesen los datos de entrada de los que disponemos y que sirvan en el futuro para estimar un resultado

fiable y generalizable a más conjuntos de datos similares.

Disponemos de información relativa a planificaciones teóricas (cada una de ellas asociada a una orden de trabajo) que son propuestas por clientes de la empresa e incluyen *tracks* GPS bien definidos en el espacio y contenedores planificados en localizaciones geográficas concretas. Estas órdenes de trabajo se diseñan de antemano para ejecutarse en la vida real con la mayor eficiencia posible. Mientras se llevan a cabo las rutas reales de los vehículos de recogida, se recoge información acerca de las mismas variables de las que se dispone información teórica.

Las trayectorias geográficas están definidas por puntos, que a su vez están definidos por sus coordenadas geográficas (longitud y latitud). Cada trayectoria está definida por numerosos puntos que se obtienen de forma diferente para las rutas teóricas y para las rutas reales. En cuanto a las rutas teóricas, estas pueden obtenerse mediante optimizadores que generan la mejor ruta según las condiciones especificadas. Para las rutas reales, los puntos son los que registra el GPS cada cierto tiempo. El tamaño de cada intervalo temporal depende de la velocidad del vehículo, el ángulo que gira en ciertos tramos y la distancia recorrida desde el último registro de posición para asegurar que la distribución de puntos registrados permita reconstruir una aproximación fiable de las rutas.

3. Preprocesamiento de datos y descripción de variables

En proyectos que involucren el análisis de información espacial, visualización y procesamiento de datos geográficamente referenciados, se emplean *Sistemas de Información Geográfica* con los que podemos obtener información gráfica como la mostrada a continuación:

Figura 1: Ejemplo de la información geográfica de una cierta orden de trabajo. En rojo, la ruta teórica planificada y en verde la ruta real ejecutada. Los puntos de color amarillo son los contenedores sí planificados y sí recogidos, los puntos de color azul son los contenedores sí planificados y no recogidos y los puntos de color gris son los contenedores no planificados y sí recogidos. La importancia de esta diferenciación se desarrollará en la siguiente subsección.

3.1. Exploración del conjunto de datos

Durante la realización de este trabajo, se ha realizado un minucioso análisis de las bases de datos de la empresa para valorar la cantidad de datos necesarios, su calidad y la disponibilidad de estos de acuerdo a todas las variables de interés. En general, es deseable disponer de un conjunto de datos con suficiente variedad como para que los modelos puedan aprender de todos los casos posibles que pueden darse en la realidad.

La información se encuentra dividida en *órdenes de trabajo*. Las órdenes de trabajo consideradas son agrupaciones de datos que contienen información acerca de la geometría de la ruta teórica planificada para ser recorrida por el vehículo de recogida, la geometría de la ruta seguida por el vehículo en la realidad al ejecutar la orden e información sobre el cumplimiento de la recogida de todos los contenedores que intervienen en la orden.

En cuanto a dichos contenedores, se ha observado que se registran hasta tres tipos de situaciones en las bases de datos:

- Contenedores **SÍ** planificados y **SÍ** recogidos: Esta es la situación óptima y deseable que se da cuando el vehículo de recogida ejecuta correctamente la planificación teórica en un cierto contenedor marcado por un identificador. Llamaremos $N(1,1)$ al número de contenedores de este tipo en una determinada orden de trabajo.
- Contenedores **SÍ** planificados y **NO** recogidos: La situación contraria a la anterior e indicadora de una mala ejecución de una orden de trabajo en un cierto contenedor, lo cual afectará negativamente a la bondad de la orden. Llamaremos $N(1,0)$ al número de contenedores de este tipo en una determinada orden de trabajo.
- Contenedores **NO** planificados y **SÍ** recogidos: Alternativamente, el vehículo puede recoger un contenedor cuyo identificador no aparezca en la planificación original, pero se registra igualmente en los datos de la ejecución de la orden de trabajo. Llamaremos $N(0,1)$ al número de contenedores de este tipo en una determinada orden de trabajo.

Atendiendo a la disponibilidad de datos, se ha hecho una selección suficientemente representativa de 623 órdenes de trabajo.

3.2. Variables de interés

Se han analizado grandes cantidades de datos reales cedidos por la empresa para seleccionar aquellas agrupaciones de datos que mejor se adaptan al objetivo de este trabajo. A partir de ellas, se han construido una serie de variables numéricas que tratan de modelizar aproximadamente las características más relevantes de una orden de trabajo manteniendo un cierto equilibrio con el número de datos utilizables para que las estimaciones sean lo más fiables posible. Estudiando la influencia que las variables tienen respecto a la valoración final que podemos realizar sobre cada orden de trabajo, se van a entrenar modelos basados en las variables definidas y descritas a continuación por orden de importancia.

3.2.1. Proporción de contenedores bien recogidos

En cada orden de trabajo, ha de priorizarse la recogida de la mayor cantidad de contenedores de entre los planificados. Si para cada orden de trabajo extraemos el número de contenedores de cada tipo, podemos definir una variable que sea la proporción de contenedores sí planificados y sí recogidos respecto del total de contenedores planificados. Así se define la variable `BienHechos`:

$$A = \frac{N(1,1)}{N(1,1) + N(1,0)} \tag{1}$$

Por construcción, es una variable que ya se encuentra normalizada entre 0 y 1.

3.2.2. Grado de similitud entre trayectorias

Dadas dos trayectorias geográficas definidas por un conjunto de puntos en el mapa, se desea calcular de algún modo su grado de similitud. Necesitamos una medida objetiva y cuantificable de cómo de parecidas son dos rutas, es decir, la *distancia* que las separa. Queremos obtener un valor numérico que especifique el nivel de semejanza entre la ruta teórica que estaba planificada para los vehículos de recogida y la verdadera ruta que ha seguido el vehículo en cada una de las órdenes de trabajo.

En general, el cálculo de la similitud de pares de trayectorias es un problema matemático complejo y que se ha seguido estudiando en los últimos años. Por ello, se han propuesto una gran cantidad de algoritmos que pueden ser adecuados en mayor o menor medida para cada problema concreto. La eficacia de estos algoritmos depende de la complejidad de las trayectorias y de sus características geométricas. En nuestro caso, puede verse en la figura 1 que las rutas teóricas y las rutas reales pueden cruzarse y tienen una forma notablemente complicada por lo que no resulta trivial hallar una medida objetiva de cómo de parecidos son dos *tracks* GPS cualesquiera. De entre muchos de los algoritmos más populares, se detalla a continuación la distancia elegida finalmente por ser la más precisa en lo referido a los requerimientos de este trabajo.

Sean T^1 y T^2 dos trayectorias, ambas definidas como un *array* de n_1 y n_2 puntos de dimensión 2 cada uno, respectivamente. Sea $P^1_{i_1}$ el punto i_1-ésimo del *array* de puntos de la trayectoria T^1 y $P^2_{i_2}$ el punto i_2-ésimo del *array* de puntos de la trayectoria T^2. Como paso previo, definimos la *Segment-Path Distance* (SPD) de la trayectoria T^1 a la trayectoria T^2 como

$$D_{SPD}(T^1, T^2) = \frac{1}{n_1} \sum_{i_1=1}^{n_1} D_{PT}(P^1_{i_1}, T^2) \tag{2}$$

donde

$$D_{PT}(P^1_{i_1}, T^2) = \min_{i_2 \in [0,\ldots,n_2-1]} D_{PS}(P^1_{i_1}, S^2_{i_2}) \tag{3}$$

siendo $D_{PS}(P^1_{i_1}, S^2_{i_2})$ la distancia del punto $P^1_{i_1}$ al segmento $S^2_{i_2}$.

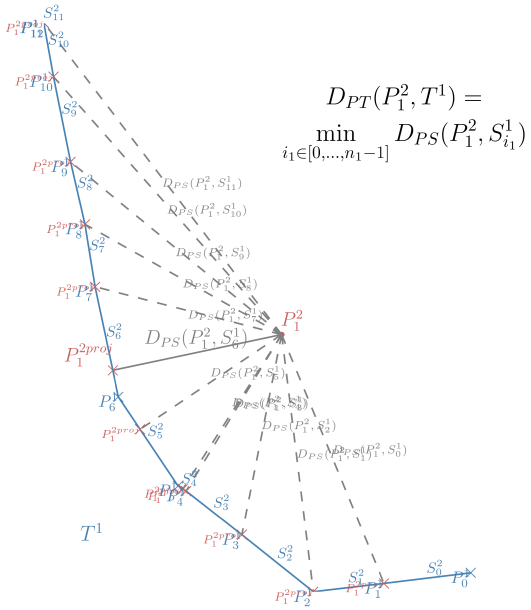

Figura 2: Cálculo de la distancia del primer punto de la trayectoria T^2 a la trayectoria T^1.

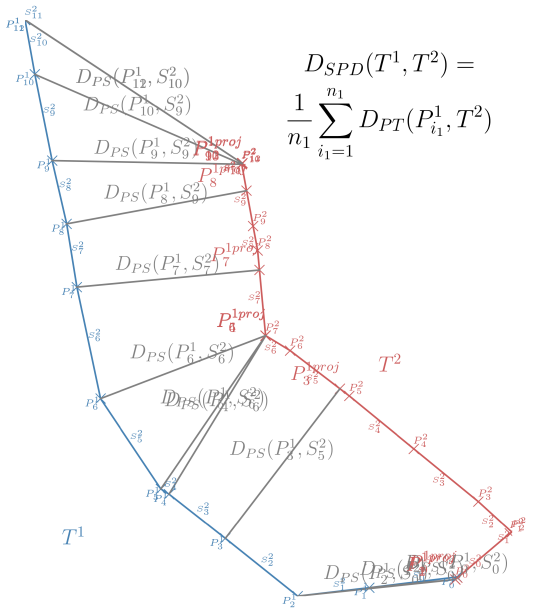

Figura 3: Distancias a considerar en el cálculo de la SPD de la trayectoria T^1 a la trayectoria T^2.

Como los puntos son coordenadas geográficas, la distancia más precisa a considerar entre dos puntos será la que viene dada por la fórmula del semiverseno:

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \tag{4}$$

donde r es el radio de la Tierra, φ_1 y φ_2 son las latitudes de los dos puntos y λ_1 y λ_2 son las longitudes de los dos puntos. Sin embargo, para reducir notablemente el tiempo de computación si han de realizarse muchos estudios de similitud, puede aproximarse por la distancia euclídea y ofrecer resultados esencialmente similares en nuestro caso.

Intuitivamente, la SPD se puede interpretar como la media de todas las distancias desde los puntos que componen la trayectoria T^1 a la trayectoria T^2. No obstante, el grado de similitud entre dos trayectorias debería ser independiente del orden en el que se considera una u otra, por lo que la simetrizamos definiendo

$$D_{SSPD}(T^1, T^2) = \frac{D_{SPD}(T^1, T^2) + D_{SPD}(T^2, T^1)}{2} \tag{5}$$

que recibe el nombre de *Symmetrized Segment-Path Distance* (SSPD) y será la que empleemos de ahora en adelante para cuantificar el grado de similitud entre parejas de rutas teóricas y rutas reales. Esta distancia permite comparar trayectorias de diferente longitud y no se ve gravemente afectada por el hecho de que el indexado temporal de los puntos de las dos trayectorias sea muy diferente. Su coste computacional es $\mathcal{O}(n_1 n_2)$.

Esta distancia es un valor entre cero e infinito. Una distancia muy pequeña indica un alto grado de similitud entre dos rutas, mientras que una distancia elevada indica que dos rutas son muy dispares. Sin embargo, para que el entrenamiento sea efectivo en los modelos de *machine learning* posteriores, debemos normalizar las variables a una escala común. Para que el grado

de similitud en función de la distancia SSPD tome valores entre 0 y 1, se puede utilizar una transformación del tipo

$$B = \frac{\xi}{\xi + D_{SSPD}(T^1, T^2)} \tag{6}$$

que es estrictamente decreciente y convierte los valores del intervalo $(0, \infty)$ en valores del intervalo $(0, 1)$. Además, siempre satisface que cuando la distancia tiende a 0, el grado de similitud tiende a 1 y que cuando la distancia tiende a infinito, el grado de similitud tiende a 0. Para elegir el parámetro ξ, hay que asegurarse de que la distribución final de la variable B es coherente con el porcentaje de las distancias que pueden considerarse buenas o malas según nuestros criterios. También podemos fijar manualmente el valor de la similitud de varias parejas de rutas y ajustar la función. En el anexo, se detalla una explicación más exhaustiva de la obtención empírica de este parámetro que podremos fijar como $\xi = 0,001$. Esta variable B será la variable `Similitud`.

3.2.3. Proporción de contenedores adicionales recogidos

Más allá de la planificación teórica, en la realidad puede producirse la recogida de algunos contenedores que no pertenecen a la ruta teórica diseñada. Esto se detecta cuando el vehículo recoge un contenedor con un identificador no perteneciente a los que figuran en la planificación de una determinada orden de trabajo. Esta situación puede darse debido a imprecisiones en el sistema GPS que guía a los conductores, calles cortadas inesperadamente, errores humanos o cambio de planes tras el diseño de las rutas debido a nuevas necesidades del contratista. Para los modelos posteriores, consideraremos que la recogida de contenedores extra no es algo necesariamente negativo si no supone una desviación excesiva respecto del *track* GPS teórico y si no perjudica la recogida de aquellos contenedores que sí se habían planificado originalmente.

Se define la variable `Adicionales` como la proporción de contenedores no planificados pero sí recogidos respecto del total de contenedores asociados a la orden de trabajo:

$$C = \frac{N(0,1)}{N(1,1) + N(1,0) + N(0,1)} \tag{7}$$

Por construcción, es una variable que ya se encuentra normalizada entre 0 y 1.

3.2.4. Desviación relativa entre longitud real y teórica

Regresando al análisis de la geometría de los *tracks* GPS teórico y real, también puede tener cierto interés la longitud de dichos *tracks*. Esta variable puede parecer no necesariamente tan fiable como la de la similitud entre rutas porque la longitud de las rutas puede verse afectada en algunos casos por situaciones imprevistas que no necesariamente reducen la calidad de la ejecución de la orden de trabajo. Esto incluye la necesidad de visitar el vertedero para descargar antes de continuar con la realización de la orden o rodeos sin especial importancia alrededor de los núcleos urbanos de interés antes de comenzar o después de finalizar la recogida de todos los contenedores previstos. No obstante, veremos que sí que es una variable de interés por su correlación con otros indicadores de bondad, aunque le asignemos el menor peso.

Calculamos el valor absoluto de la diferencia entre la longitud de la ruta teórica y la longitud de la ruta real, dividida entre la longitud de la ruta teórica:

$$D^* = \frac{|Longitud\ Teorica - Longitud\ Real|}{Longitud\ Teorica} \tag{8}$$

Matemáticamente, esta variable no está directamente normalizada entre 0 y 1 para cualquier par de longitudes. No obstante, se ha observado que la distribución de esta variable no se aleja demasiado de dicho rango y es sencillo trasladar los valores al rango deseado sin alterar la forma de la distribución mediante la transformación lineal

$$D = \frac{D^* - D^*_{min}}{D^*_{max} - D^*_{min}} \tag{9}$$

que asocia el valor 0 a D^*_{min} (mínimo valor de D^*) y el valor 1 a D^*_{max} (máximo valor de D^*), así como valores intermedios a todos los demás valores de D^*. A esta variable normalizada D la llamaremos `RatioLongitudes`.

3.2.5. Bondad

Como variable dependiente, buscamos un indicador que nos dé una idea de cómo de bien ha sido ejecutada una determinada orden de trabajo. Como es de esperar, no existe una única forma universal de cuantificar la bondad asociada a una orden de trabajo. Esto depende del nivel de importancia que cada contratista quiera asignar a cada una de las variables.

El conjunto de datos con el que estamos trabajando no tiene asociadas unas bondades específicas a partir de las cuales trabajar. Por tanto, vamos a simular nuestras propias bondades con unos pesos específicos para cada variable. Asumimos que el orden de importancia de las variables es el orden en el que se han descrito en los apartados anteriores (de mayor a menor). Supongamos que un cierto contratista quiere asignar, en función de sus intereses, los siguientes pesos relativos: $(10\ ,\ 4\ ,\ 2\ ,\ -1)$. Construyamos una combinación lineal de las cuatro variables con pesos 10, 4, 2 y -1 que reflejan el nivel de importancia que se le podría dar a cada variable, es decir,

$$10A + 4B + 2C - D \tag{10}$$

donde la variable D (desviación relativa entre longitud teórica y real) lleva signo negativo porque una buena orden de trabajo debe minimizarla y no maximizarla como las demás. Esta combinación lineal nos dice que a la peor orden de trabajo posible (con $A = 0$, $B = 0$, $C = 0$ y $D = 1$) se le asignaría un valor

$$10 \cdot 0 + 4 \cdot 0 + 2 \cdot 0 - 1 = -1 \tag{11}$$

mientras que a la mejor orden de trabajo posible (con $A = 1$, $B = 1$, $C = 1$ y $D = 0$) se le asignaría un valor

$$10 \cdot 1 + 4 \cdot 1 + 2 \cdot 1 - 0 = 16 \tag{12}$$

No obstante, las variables A, B, C y D son, en general, números reales no enteros y una persona humana probablemente asignaría bondades enteras *a ojo* a cada orden de trabajo con el fin

de disponer de datos con los que entrenar los modelos. Asimismo, dicha valoración por parte de una persona traería consigo un cierto error humano que vamos a modelizar mediante ruido gaussiano.

Consideremos una variable aleatoria normal de media $\mu = 0$ y desviación típica $\sigma = 0{,}7$. Al sumar esta variable a la combinación descrita en (10), puede llegar a agrandarse el rango $[-1, 16]$ que podía alcanzarse con la combinación (10). Para un mejor entendimiento de las bondades, aplicamos de nuevo una transformación lineal análoga a la descrita en la expresión (9), pero multiplicada por un factor 10 para ajustar estas bondades al rango $[0, 10]$.

El siguiente paso es realizar una división de estas bondades en 4 categorías (o *clases*) con las que poder clasificar las órdenes de trabajo según sean ***malas*** (clase 0), ***regulares*** (clase 1), ***razonablemente buenas*** (clase 2) o ***muy buenas*** (clase 3). Con esto podremos entrenar modelos de clasificación que predigan automáticamente a qué grupo pertenece una orden de trabajo nueva. Dado el rango de valores de bondad que disponemos entre 0 y 10, vamos a realizar un *clustering* mediante el algoritmo k-`means`. Este es el primer algoritmo de *machine learning* que se va a poner en práctica en este trabajo y pertenece al grupo de algoritmos de aprendizaje no supervisado porque será capaz de hallar $k = 4$ grupos o *clusters* a partir de datos no etiquetados. Inicialmente, se eligen $k = 4$ *centroides* de forma aleatoria. Los puntos del conjunto se asocian a un *cluster* concreto según su centroide más cercano. En cada iteración se actualizan los centroides pasando a ser la media de los puntos que conforman cada *cluster*. Se procede así hasta conseguir minimizar una función de coste dada por

$$J = \sum_{j=1}^{k} \sum_{b_i \in S_j} ||b_i - c_j||^2 \tag{13}$$

donde $k = 4$ es el número de *clusters*, S_j es cada uno de los *clusters* S_0, S_1, S_2 y S_3, b_i es cada uno de nuestros puntos que representan bondades y c_j son los centroides. La partición óptima es la que posee centroides c_j tal que J es mínima. En nuestro caso, esta partición sera unidimensional en el rango de bondades comprimidas en el intervalo [0,10]. Usaremos este método para obtener una idea de cómo podría realizarse una división objetiva en las 4 clases en las que querremos clasificar las órdenes de trabajo.

Así, obtenemos la siguiente partición en cuatro intervalos reales.

Intervalos de bondades reales	**Clase**
$[0'00, 4'45)$	0 (malas)
$[4'45, 7'27)$	1 (regulares)
$[7'27, 8'35)$	2 (razonablemente buenas)
$[8'35, 10'00]$	3 (muy buenas)

Tabla 1: Partición en 4 clases según la bondad.

Finalmente, redondeamos las bondades a un número entero para dar más realismo a las notas de bondad que asignaría una persona humana y las incluimos en uno de los cuatro intervalos. En términos de números enteros, la división queda

Bondades enteras	Clase
$0, 1, 2, 3, 4$	0 (malas)
$5, 6, 7$	1 (regulares)
8	2 (razonablemente buenas)
$9, 10$	3 (muy buenas)

Tabla 2: Asignación de bondades enteras a cada clase.

3.2.6. Revisión conjunta

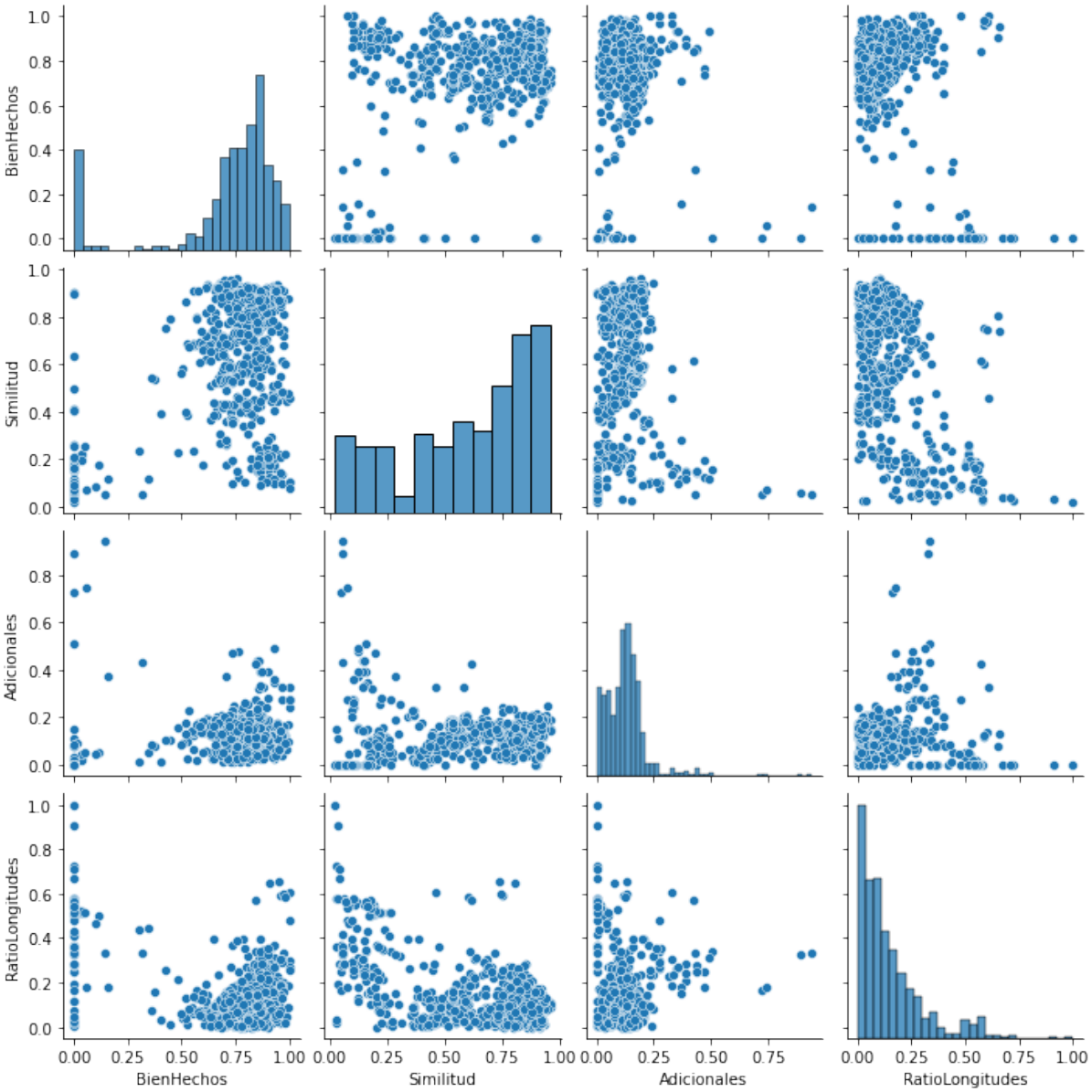

Figura 4: En la diagonal, distribución de los datos de cada variable. Fuera de la diagonal, representaciones gráficas de las variables tomadas dos a dos.

Vemos que la proporción de contenedores `BienHechos` suele agruparse en torno a valores altos, aunque también presenta una barra aislada en cero debido a la presencia de órdenes de trabajo con nulo cumplimiento tal y como están almacenadas. Esto puede deberse en muchos casos a que los vehículos de recogida no cuentan con sistemas de identificación que permitan llevar a cabo el registro de contenedores a pesar de que pueda existir una alta similitud entre la ruta planeada y la ruta real. No obstante, los modelos nos permitirán señalar este tipo de anomalías.

Aquí podemos observar ya la correlación que existe en algunos casos ya que vemos zonas especialmente densas. Parece haber una relación entre alta proporción de contenedores bien recogidos, baja proporción de contenedores adicionales recogidos y baja desviación relativa entre la longitud de la ruta real y de la teórica. Por otro lado, como era de esperar, hay una correlación entre la alta proporción de contenedores bien recogidos y el alto grado de similitud, pero menos clara porque también hay una cierta cantidad de órdenes con alto grado de cumplimiento pero baja similitud debido a cambios de última hora en los planes del contratista o el descarte de la planificación teórica acordada originalmente.

Asimismo, se ha obtenido que la distribución de bondades es la siguiente:

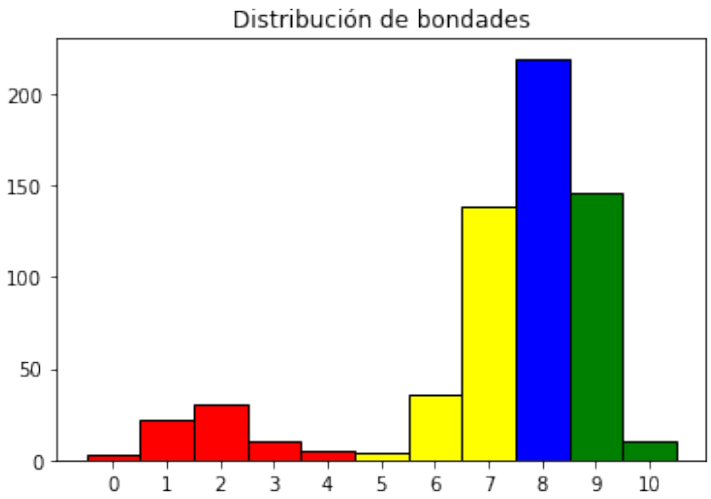

Figura 5: Cantidad de órdenes de trabajo con cada valor de bondad del 0 al 10. En rojo, órdenes ***malas***. En amarillo, órdenes ***regulares***. En azul, órdenes ***razonablemente buenas***. En verde, órdenes ***muy buenas***.

4. Técnicas estadísticas relevantes en *machine learning*

Más allá de un simple vistazo a los resultados obtenidos, podemos recurrir a ciertas técnicas estadísticas como las dos que se describen a continuación para evaluar la capacidad de los modelos.

4.1. Curvas ROC y matrices de confusión

En problemas de clasificación, se hace uso de las curvas ROC (*Receiver Operating Characteristic*) para evaluar la eficacia de los modelos. Las curvas ROC son representaciones gráficas de probabilidades y el área bajo la curva (AUC, por sus siglas en inglés) sirve para medir el grado de separabilidad entre las distintas clases que componen la salida del modelo.

Supongamos que el modelo en cuestión es de clasificación binaria, es decir, debe predecir a cuál de las dos clases pertenece una cierta muestra. Denominemos a una de las dos clases como la *clase positiva* y a la otra como la *clase negativa*. Al intentar realizar una predicción acerca de la clase a la que pertenece una cierta muestra, pueden darse cuatro situaciones:

- Verdadero positivo (VP): El modelo predice correctamente la clase positiva.
- Falso positivo (FP): El modelo predice incorrectamente la clase positiva.
- Falso negativo (FN): El modelo predice incorrectamente la clase negativa.
- Verdadero negativo (VN): El modelo predice correctamente la clase negativa.

Haciendo que el modelo realice predicciones sobre un conjunto de datos, podríamos construir una *matriz de confusión* que refleje el número de predicciones de cada tipo. Para el caso de la clasificación binaria, esta sería una matriz 2×2 de la forma

$$\begin{pmatrix} VP & FP \\ FN & VN \end{pmatrix} \tag{14}$$

Asimismo, se definen la tasa de verdaderos positivos (TVP), la tasa de falsos positivos (TFP) y la tasa de verdaderos negativos (TVN) como

$$TVP = \frac{VP}{VP + FN} \qquad TFP = \frac{FP}{FP + VN} \qquad TVN = \frac{VN}{FP + VN} \tag{15}$$

Nótese que $TVN = 1 - TFP$. A la TVP también se le conoce comúnmente como *sensibilidad* y a la TVN como *especificidad*. Las curvas ROC son representaciones bidimensionales de la TVP (sensibilidad) frente a la TFP (1−especificidad) que se obtienen al evaluar el modelo. No obstante, nuestros modelos no serán de clasificación binaria, sino de clasificación de cuatro clases. Por tanto, para cada una de las cuatro clases, debe reducirse el problema a un problema de clasificación de *una contra el resto* y se obtiene una curva para cada clase.

4.2. *Bootstrapping*

Los métodos basados en *bootstrapping* nos permiten obtener una aproximación adecuada de los intervalos de confianza asociados a los resultados de los modelos de *machine learning* para conocer su capacidad. Un intervalo de confianza ofrece información acerca del rango en el que reside la capacidad real de un modelo para producir resultados fiables.

En cada iteración del método *bootstrap*, disponemos de un subconjunto de datos de entrenamiento y un subconjunto de datos de test diferente. Este conjunto de datos de entrenamiento tiene el mismo tamaño que el conjunto de datos original y ha sido creado mediante la extracción

aleatoria de elementos uno a uno con reemplazo, por lo que puede contener muestras repetidas. Las muestras del conjunto de datos que no han sido elegidas pasan a formar parte del conjunto de test. Se entrena un cierto modelo con esta selección de datos de entrenamiento y se evalúa su capacidad sobre esta selección de datos de test mediante un cierto estadístico como puede ser, por ejemplo, la proporción de aciertos en un modelo clasificador. Este proceso se repite M veces formando cada vez subconjuntos de entrenamiento diferentes con muestras que pueden estar repetidas, entrenando el correspondiente modelo y evaluando su capacidad M veces en total. Así, llegamos finalmente a una distribución formada por M datos cuya forma podemos estudiar para determinar un intervalo de confianza al porcentaje deseado.

Para modelos de clasificación, el estadístico a medir puede ser la precisión y para modelos de regresión, el estadístico a tener en cuenta puede ser una métrica de error como el error cuadrático medio. En el desarrollo de los modelos, obtendremos los intervalos de confianza al 95 %.

5. Árboles de decisión

Los árboles de decisión son modelos de predicción basados en la creación de diagramas que representan decisiones lógicas. Se trata de una técnica de aprendizaje supervisado a la que suministramos las cuatro variables diseñadas anteriormente (variables independientes) y una cierta variable objetivo (variable dependiente). Dado que los árboles de decisión pueden emplearse como modelos clasificadores o como regresores, dicha variable objetivo será la bondad en su forma cualitativa en el primer caso (clase 0, 1, 2, ó 3), o en su forma cuantitativa en el segundo caso (bondades enteras del 0 al 10). Se evaluará la eficacia de ambos tipos de algoritmos. En el primer caso, se aprovecha la utilidad gráfica que presentan los clasificadores en el caso de necesitar tan sólo una valoración cualitativa de una orden de trabajo. En el segundo caso, se aprovecha la utilidad numérica que puede suponer proporcionar una valoración cuantitativa.

Un árbol está formado por diversos *nodos* unidos por *ramas*. Cada nodo representa una cierta proposición lógica y cada rama representa la decisión tomada en una cierta dirección. El desarrollo comienza en un *nodo raíz* y, a partir de ahí, se realizan sucesivas bifurcaciones de acuerdo a la partición más óptima del conjunto de datos. El desarrollo del árbol culmina en los llamados *nodos hoja* donde ya no se evalúa ninguna proposición lógica.

Matemáticamente, disponemos de un conjunto de vectores de entrenamiento $x_i \in \mathbb{R}^n,\ i = 1, \ldots, l$ donde n es el número de variables independientes y l es el número de vectores de entrenamiento. Además, disponemos de un vector $y \in \mathbb{R}^l$ que define los valores de la variable dependiente (etiqueta de cada clase para clasificación o valores numéricos de la bondad para regresión). La idea es realizar sucesivas particiones que agrupen datos con la misma etiqueta de clase en clasificación o valores numéricos de bondad parecidos en regresión. Sea Q_m el conjunto de datos en el nodo m que tendrá N_m datos. Llamamos $\theta = (j, t_m)$ a la bifurcación que discrimina mediante la variable j con umbral t_m. Entonces, a partir del conjunto Q_m de datos del nodo m se definen dos nuevos conjuntos tras la bifurcación:

$$Q_m^{izda}(\theta) = \{(x_i, y_i) | x_{ij} \leq t_m\} \tag{16}$$

$$Q_m^{dcha}(\theta) = Q_m \backslash Q_m^{izda}(\theta) \tag{17}$$

Utilizando una función de impureza (clasificación) o error (regresión) que llamaremos H, se define la función G en cada nodo tal que

$$G(Q_m, \theta) = \frac{N_m^{izda}}{N_m} H(Q_m^{izda}(\theta)) + \frac{N_m^{dcha}}{N_m} H(Q_m^{dcha}(\theta)) \tag{18}$$

Los parámetros que definen una cierta bifurcación son los que minimizan dicha función:

$$\theta^* = \text{argmin}_\theta \; G(Q_m, \theta) \tag{19}$$

Y el proceso continúa generando nuevos conjuntos $Q_m^{izda}(\theta^*)$ y $Q_m^{dcha}(\theta^*)$ hasta alcanzar los nodos terminales. Las funciones H se mostrarán explícitamente en los apartados posteriores dependiendo de si se quieren utilizar árboles de decisión como clasificadores o como regresores.

En cualquier caso, el primer paso será hacer una partición aleatoria del conjunto de datos en dos subconjuntos:

- Datos de entrenamiento: Se utilizarán para que el modelo aprenda las características más descriptivas. De las 623 órdenes de trabajo disponibles, tomaremos el 70 % para entrenar, es decir, 436 órdenes de trabajo.
- Datos de test: Se utilizarán para poner a prueba la eficacia del modelo tras su entrenamiento. De las 623 órdenes de trabajo disponibles, tomaremos el 30 % para validar, es decir, 187 órdenes de trabajo.

5.1. Árboles clasificadores

Como medida de la homogeneidad de un conjunto de datos, vamos a utilizar la *impureza de Gini* que tiene la forma:

$$H(Q_m) = 1 - \sum_k p_{mk}^2 \tag{20}$$

donde p_{mk} es la proporción de órdenes de trabajo de la clase k en el nodo m. Se trata de un índice entre 0 y 1 tal que el valor mínimo se alcanza cuando el conjunto de datos de órdenes de trabajo en el nodo m es perfectamente homogéneo (todas las órdenes tienen una misma clase) y el valor máximo se alcanza cuando la heterogeneidad es máxima.

5.1.1. Representación de un árbol

A continuación, se muestra la representación gráfica de uno de los árboles de decisión que genera el entrenamiento de un clasificador.

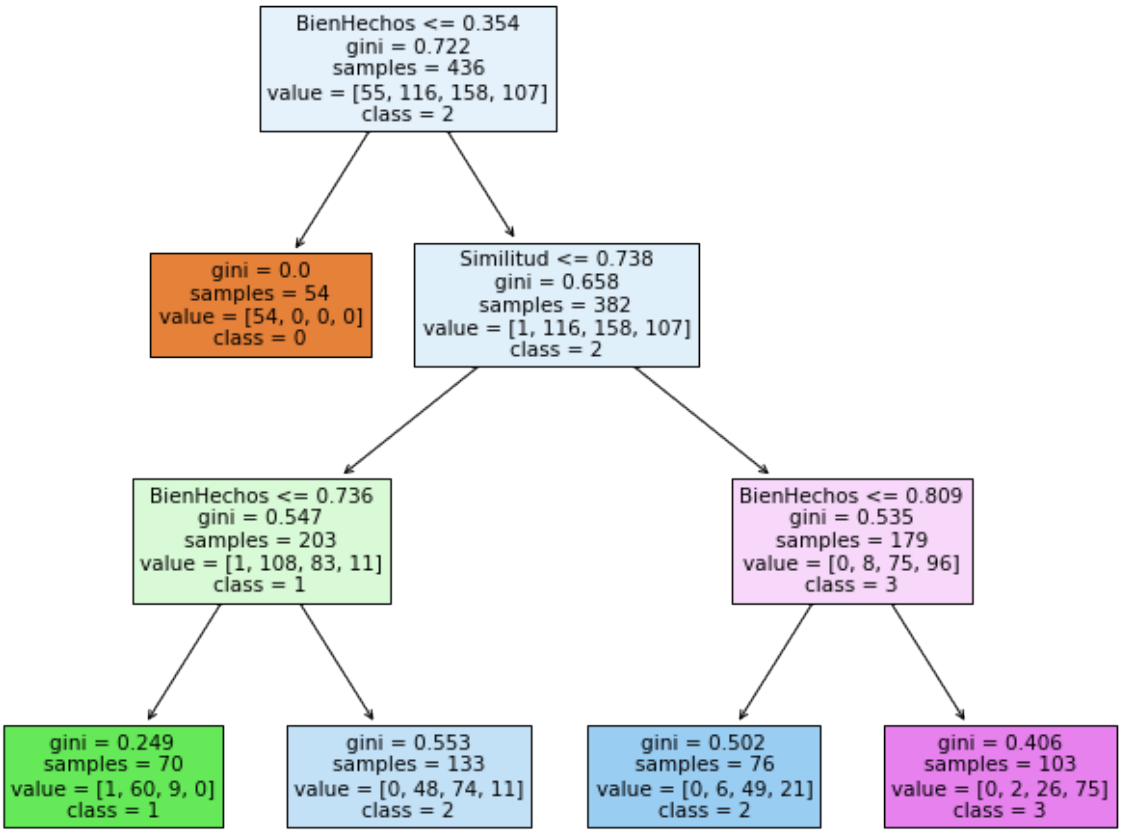

Figura 6: Un árbol de decisión con profundidad máxima 3.

donde la clase 0 se refiere a las órdenes ***malas***, la clase 1 a las órdenes ***regulares***, la clase 2 a las órdenes ***razonablemente buenas*** y la clase 3 a las órdenes ***muy buenas***.

En realidad, un árbol completo posee numerosas ramificaciones adicionales hasta que la impureza de Gini sea nula en todos los nodos hoja. Además, si permitiéramos un mayor desarrollo, observaríamos condiciones en función de más variables ya que aquí sólo han aparecido dos de ellas. Sin embargo, resulta contraproducente desarrollar el árbol hasta los nodos hoja sin hacer una poda (*pruning*) porque se pierde totalmente la legibilidad y se corre el riesgo de sobreajustar el modelo y que no pueda funcionar bien para otros conjuntos de datos más allá del de entrenamiento.

En particular, a partir del árbol de la figura 6, se deduce la siguiente información:

Condiciones	**Clase**
$BienHechos \leq 0'354$	0
$BienHechos \in (0'354, 0'736]$ y $Similitud \leq 0'738$	1
$BienHechos > 0'736$ y $Similitud \leq 0'738$	2
$BienHechos \in (0'354, 0'809]$ y $Similitud > 0'738$	2
$BienHechos > 0'809$ y $Similitud > 0'738$	3

Tabla 3: Interpretación del árbol de decisión.

Y al poner a prueba el modelo para que intente predecir la clase a la que pertenecen las muestras del conjunto de test, se obtiene la siguiente matriz de confusión:

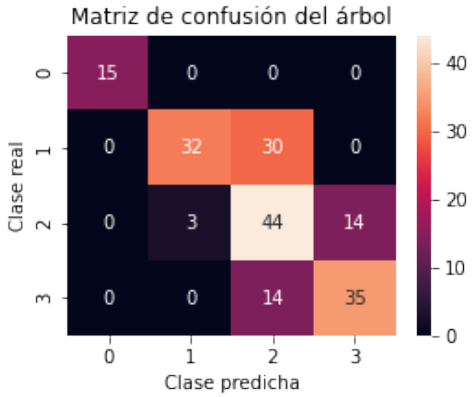

Figura 7: Matriz de confusión del árbol de decisión comparando las clases predichas por el modelo frente a las reales suministradas en el subconjunto de datos de test.

El número de aciertos del modelo es la suma de los valores de la diagonal. Por tanto, la precisión o proporción de aciertos es la traza de la matriz dividida entre la suma de todos los valores de la matriz. En este caso, dicha precisión total resulta ser del $67,4\,\%$. Un vistazo a la matriz nos permite observar que la detección de órdenes de trabajo de la clase 0 (***malas***) es perfecta, lo cual es especialmente relevante para detectar con eficacia anomalías graves en ciertas órdenes de trabajo. Por el contrario, vemos que el modelo predice demasiadas veces la clase 2 (órdenes ***razonablemente buenas***) cuando la orden de trabajo no pertenece realmente a dicha clase, aunque las desviaciones son a las clases colindantes.

Podemos generar tantos árboles de decisión como queramos según la partición aleatoria del conjunto de datos. En general, deberíamos esperar árboles que proporcionen una información similar incluso si esta información se presenta en distinto orden. No obstante, puede haber semillas concretas que den lugar a una clasificación inesperada con las otras variables que no han aparecido en el árbol de la figura 6. Para no tener que depender de la información de un solo árbol, se recurre a otras técnicas como los bosques aleatorios.

5.1.2. Bosque aleatorio

Se trata de una técnica basada en la construcción de tantos árboles como se quiera, a partir de conjuntos de datos extraídos aleatoriamente. Finalmente, se agrupan las predicciones de cada uno de los árboles para realizar una predicción final. Es por ello que entra dentro del conjunto de métodos de *ensemble* y puede ayudar a reducir la alta variabilidad que a menudo presentan los árboles de decisión individuales. Si seguimos imponiendo una profundidad máxima de hasta la tercera generación, el problema reside en hallar el número óptimo de árboles que deben extraerse aleatoriamente para maximizar la precisión del modelo sin perjudicar gravemente el coste computacional.

En este caso, se ha determinado que para el caso de 70 estimadores, se llega a obtener una precisión de hasta el $76,5\,\%$, tal y como se deduce a través de la siguiente matriz de confusión:

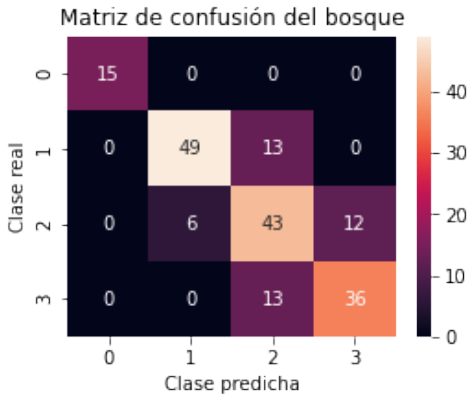

Figura 8: Matriz de confusión del bosque aleatorio comparando las clases predichas por el modelo frente a las reales suministradas en el subconjunto de datos de test.

donde se observa una mejora sustancial en la clasificación correcta de las órdenes de clase 1 que antes se clasificaban en la clase 2, incrementándose también la proporción de aciertos en las órdenes para las que se predijo la clase 2.

Apliquemos ahora el método de *bootstrapping* para hallar un intervalo de confianza al 95 %. Entrenamos el modelo 1000 veces mediante conjuntos de datos construidos a través de extracciones aleatorias de los datos originales uno a uno hasta formar conjuntos de datos del mismo tamaño que el conjunto de datos original. Así se obtiene el siguiente histograma de precisiones (proporción de aciertos del clasificador):

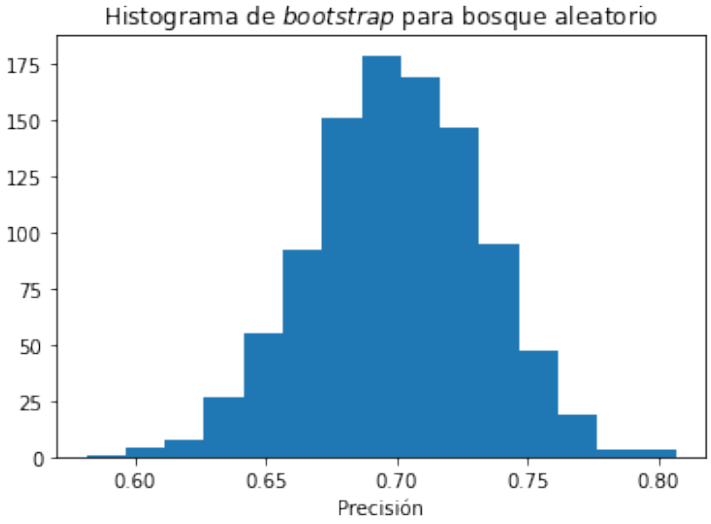

Figura 9: Distribución de precisiones para el bosque aleatorio.

El 95 % de los valores centrales del histograma se sitúan en el intervalo $[0'635, 0'762]$ por lo que concluimos que la capacidad real del modelo se sitúa entre el $63,5\,\%$ y el $76,2\,\%$ con una confianza del 95 %. Como es de esperar, este intervalo está por debajo de la precisión de $76,5\,\%$ obtenida antes ya que era un valor máximo, pero no el más habitual para este modelo.

Asimismo, podemos calcular el grado de importancia de cada variable en el modelo de acuerdo

al criterio de la *importancia de Gini* teniendo en cuenta la contribución de cada variable a la disminución de la impureza de Gini.

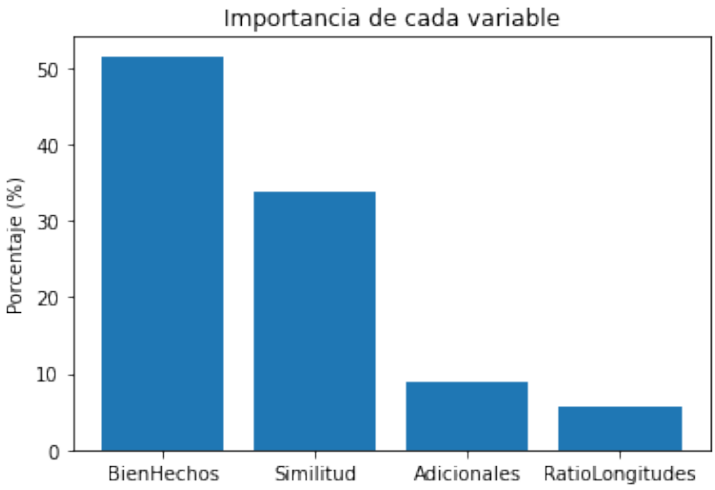

Figura 10: Grado de importancia de cada una de las 4 variables en tanto por ciento.

donde se observa que, debido a la complejidad del conjunto de datos y las correlaciones existentes entre las variables, el porcentaje de importancia de cada variable no es necesariamente proporcional al peso que le habíamos asignado originalmente en la construcción de la bondad, como cabe esperar.

Además, graficamos las curvas ROC para este clasificador:

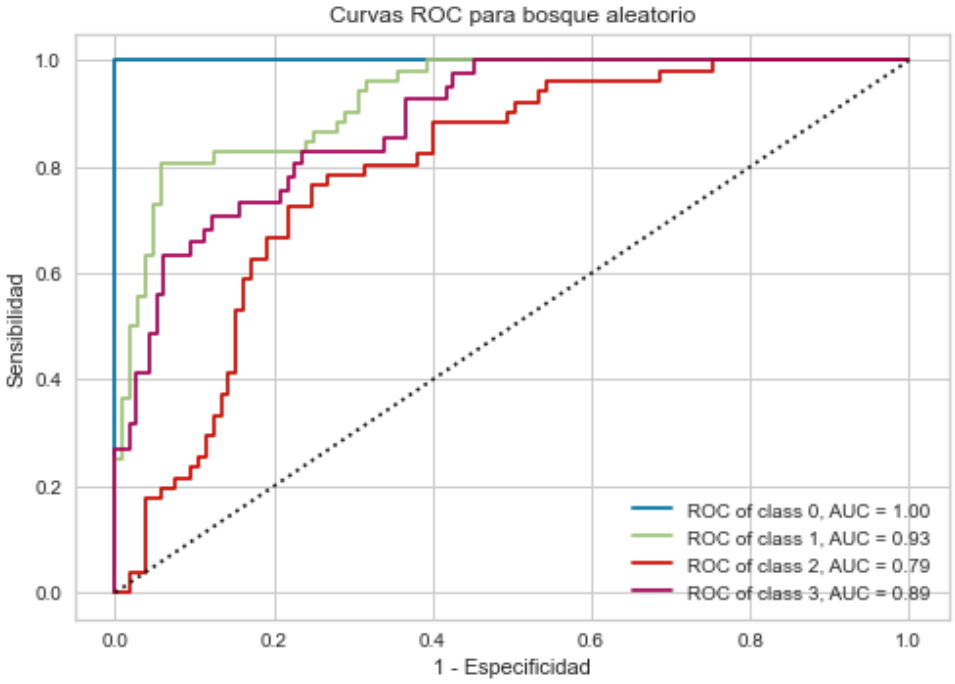

Figura 11: Curvas ROC para cada clase y área bajo cada curva (AUC).

evidenciando que la clasificación de la clase 0 se lleva a cabo perfectamente, mientras que la más problemática es la clase 2 por ser la más cercana a la diagonal y encerrar el menor área.

5.2. Árboles regresores

En este caso, la función que se pretende minimizar en cada nodo es el *error cuadrático medio*:

$$H(Q_m) = \frac{1}{N_m} \sum_{y_j \in Q_m} (y_j - \overline{y}_m)^2 \tag{21}$$

donde

$$\overline{y}_m = \frac{1}{N_m} \sum_{y_j \in Q_m} y_j \tag{22}$$

Ahora utilizaremos las bondades en su forma cuantitativa (números enteros entre el 0 y el 10) en lugar de su forma cualitativa en 4 clases. Emplearemos directamente algoritmos regresores basados en bosques aleatorios con una selección de parámetros equivalente a la del apartado anterior. La medida de cómo de buena ha sido la regresión nos la da el coeficiente R^2 definido como

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2} \tag{23}$$

donde y_i son las bondades numéricas del conjunto de test, $\overline{y}$ su media e $\hat{y}_i$ las predicciones.

En este caso, se ha obtenido $R^2 = 0,8889$, valor cercano a 1, lo que significa un buen ajuste a los valores etiquetados. Finalmente, se compara a continuación el parecido existente entre la distribución de bondades enteras etiquetadas del conjunto de test y las predicciones del modelo regresor (bondades no enteras, aunque con *bins* ajustados a los números enteros).

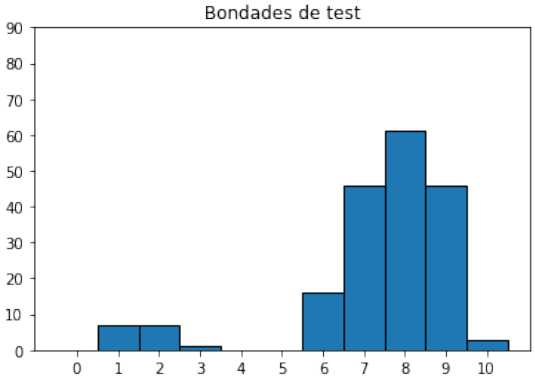

Figura 12: Bondades etiquetadas en el conjunto de test.

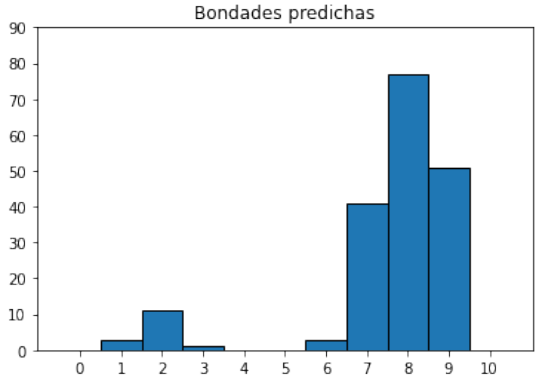

Figura 13: Predicciones del bosque regresor.

6. Redes neuronales

Además de los modelos de árboles de decisión, vamos a analizar la utilidad de las redes neuronales para aprender funciones predictoras. En particular, se aplicarán algoritmos de redes

neuronales supervisadas dado que conocemos las variables independientes y la variable objetivo con las que entrenar. Vamos a probar modelos de *perceptrón multicapa*, una red neuronal compuesta por múltiples capas en la que las conexiones entre neuronas no forman ciclos.

En primer lugar, se tiene una *capa de entrada* con tantas entradas como variables tenga el modelo. Las capas intermedias reciben el nombre de *capas ocultas* que toman la información de la capa anterior y la procesan antes de transmitirla a la capa siguiente. Finalmente, la información llega a la *capa de salida* y la transforma en el valor de salida de nuestro modelo.

Situémonos en la primera capa oculta y llamemos i a la neurona cuya dinámica queremos estudiar. Cada conexión de la red va a tener asociado un cierto peso de forma que llamaremos w_{ij} al peso asociado a la conexión $j \longrightarrow i$. Estos pesos son coeficientes que pueden adaptarse dentro de la red e indican el nivel de relevancia asignado a la información que procede de una neurona de la capa anterior. La asignación de pesos se puede realizar mediante algoritmos de *descenso de gradiente* y *retropropagación*. Empecemos por la primera capa oculta.

En primer lugar, se calcula la variable

$$z_i^{(1)} = \sum_j w_{ij}^{(1)} x_j + b_i^{(1)} \tag{24}$$

Es decir, una combinación lineal de las variables de entrada ponderada con los pesos y corregida por el coeficiente b_i que se conoce como sesgo (*bias*).

En segundo lugar, esta variable se hace pasar por una cierta función de activación que determina la información que se enviará finalmente a la capa siguiente. Por ejemplo, para problemas de clasificación, algunas funciones de activación típicas son

$$\phi_1(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \qquad \phi_2(z) = \frac{1}{1 + e^{-z}} \qquad \phi_3(z) = \text{máx}(0, z) \tag{25}$$

que son la tangente hiperbólica, la sigmoide y la ReLU (*Rectified Linear Unit*), respectivamente. Así pues, la salida de la neurona i de la primera capa oculta sería la activación

$$a_i^{(1)} = \phi^{(1)}\left(z_i^{(1)}\right) = \phi^{(1)}\left(\sum_j w_{ij}^{(1)} x_j + b_i^{(1)}\right) \tag{26}$$

Si asumimos que hay n capas ocultas, la información se transmite capa tras capa de manera que la neurona i de la capa n-ésima transmitiría la siguiente activación:

$$a_i^{(n)} = \phi^{(n)}\left(\sum_j w_{ij}^{(n)} a_j^{(n-1)} + b_i^{(n)}\right) \tag{27}$$

En general, se puede aplicar una función ϕ distinta para cada capa oculta. En nuestro caso, conocido el tamaño del conjunto de datos y la dimensión del problema, será suficiente con considerar una capa oculta cuya función de activación será una de las descritas en (25).

6.1. Redes clasificadoras

En nuestro caso de estudio, se trata de crear un clasificador que prediga a cuál de las cuatro clases diseñadas pertenece una orden de trabajo. Para ello, necesitaremos cuatro neuronas en

la capa de salida. Para problemas de clasificación en más de dos clases, necesitamos en la capa de salida una función de activación más avanzada que determine con suficiente precisión la probabilidad de que una orden de trabajo pertenezca realmente a una determinada clase. Esta información se nos proporciona a través de la función `Softmax`. Dado un vector $\mathbf{z} = (z_1, \ldots, z_k)$, la función `Softmax` comprime los valores en el intervalo [0,1] obteniéndose los siguientes valores transformados:

$$\text{Softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \tag{28}$$

Se trata de una generalización de la función sigmoide. El valor k sería el número de clases (en nuestro caso, cuatro) y se obtiene la probabilidad de que una muestra pertenezca a cada una de las cuatro clases. En principio, el resultado final de la red sería aquella clase que tenga asignada la mayor probabilidad de todas.

No obstante, después de la obtención de tal vector de probabilidades, se define una función de pérdida (o de coste). Una función de coste típicamente utilizada es la *entropía cruzada*. Se define esta función como

$$H(p, q) = -\sum_{k} p(k) \cdot \log(q(k)) \tag{29}$$

donde $p(k)$ es la probabilidad de obtener la clase k en la variable objetivo que hemos suministrado para el entrenamiento y $q(k)$ es la probabilidad de obtener la clase k según se ha calculado tras aplicar la función `Softmax`.
Esta pérdida valdría 0 para una predicción perfecta e infinito para una predicción totalmente opuesta a la esperada. Adicionalmente, a esta función se le puede sumar un término de regularización de la forma $\alpha||W||_2^2$ donde $||W||_2^2$ es una suma de pesos de la red al cuadrado y α es un parámetro que podemos ajustar. Así, los pesos grandes tendrán mayor penalización en la función de coste si α es grande y, análogamente, el efecto de regularización será menor si α es menor.

Vamos a entrenar primero un perceptrón multicapa a través de la implementación de la librería `scikit-learn`. Las redes neuronales presentan la dificultad de encontrar una configuración óptima al haber una gran variedad de parámetros que se pueden ajustar. Vamos a probar diferente número de neuronas en la capa oculta (4, 5 ó 6), funciones de activación (tangente hiperbólica, sigmoide o ReLU) y parámetros de regularización en diferentes órdenes de magnitud (10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}). Asimismo, podemos imponer el uso del resolvedor `L-BFGS` debido a que el conjunto de datos no es especialmente grande y se espera una convergencia más rápida para conjuntos de poco tamaño.

Como veremos, las redes neuronales aplicadas a nuestro conjunto de datos son más inestables que los bosques aleatorios en el sentido de que existe mayor variabilidad en los resultados. Diferentes ejecuciones de un mismo algoritmo pueden dar lugar a resultados algo diferentes y a señalar que la arquitectura óptima de la red puede ser distinta según la ejecución. No obstante, tras un análisis de los resultados más habituales que arroja la red, una combinación notablemente óptima es la de 4 neuronas en la capa oculta, función de activación ReLU y parámetro de regularización 10^{-4}. Con esta configuración, entrenamos la red y obtenemos una precisión del 77 % como también se ve en la siguiente matriz de confusión:

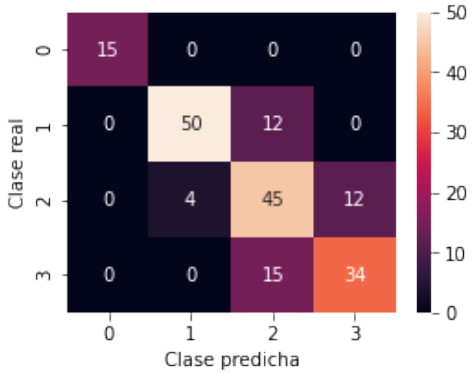

Figura 14: Matriz de confusión de la red neuronal comparando las clases predichas por el modelo frente a las reales suministradas en el subconjunto de datos de test.

Asimismo, el histograma obtenido aplicando *bootstrapping* con 2000 entrenamientos es

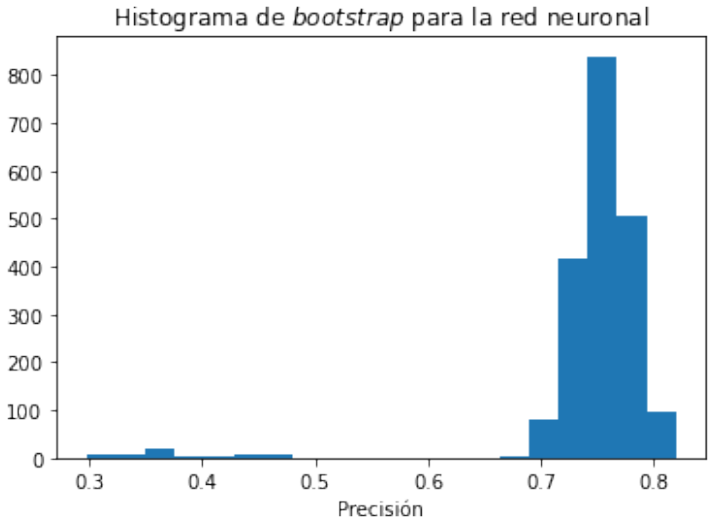

Figura 15: Distribución de precisiones para la red neuronal.

donde el 95 % de las precisiones centrales obtenidas se encuentra en el intervalo $[0'46, 0'80]$. Por tanto, la capacidad real del modelo es la definida por una proporción de aciertos en clasificación de entre el 46 % y el 80 % con una confianza al 95 %. Claramente, esta incertidumbre es mucho mayor que en los modelos estudiados anteriormente ya que, ocasionalmente, podríamos encontrar ejecuciones que dan lugar a una baja precisión por la aleatoriedad de ciertos elementos de la red. Aun así, el histograma anterior nos da una idea de que la precisión del modelo se encontrará casi siempre entre el 70 % y el 80 %.

Vamos a probar también el diseño de una red mediante `TensorFlow`. A pesar de que el conjunto de datos no es especialmente grande, vamos a probar la implementación del resolvedor `Adam`. Esto requiere el ajuste de al menos cuatro hiperparámetros descritos en el anexo de los que depende notablemente la calidad de los resultados. Probando diferentes combinaciones, puede llegarse a la conclusión de que una combinación adecuada es $\alpha = 0,07$, $\beta_1 = 0,9$, $\beta_2 = 0,999$ y $\varepsilon = 10^{-7}$. El modelo se entrena eligiendo un cierto *batch size* y un cierto número de *épocas*. El *batch size* es el número total de muestras de entrenamiento en cada *batch*, siendo un *batch*

cada uno de los grupos en los que se divide el conjunto de datos para no hacer pasar todos los datos por la red a la vez. Por otro lado, una época pasa cuando un conjunto de datos atraviesa la red hacia delante y hacia atrás una vez. A mayor número de épocas, mayor número de veces se actualizan los parámetros de la red. Se han de elegir unos números que den lugar a un descenso de la pérdida y un aumento de la precisión a unos ritmos razonables y manteniendo aproximadamente la monotonía salvo oscilaciones pequeñas. Por ello, se han elegido 80 épocas y un *batch size* de 50. Tras entrenar el modelo, podemos observar las tendencias esperadas en las curvas de la pérdida y de la precisión al validar con el conjunto de datos de test.

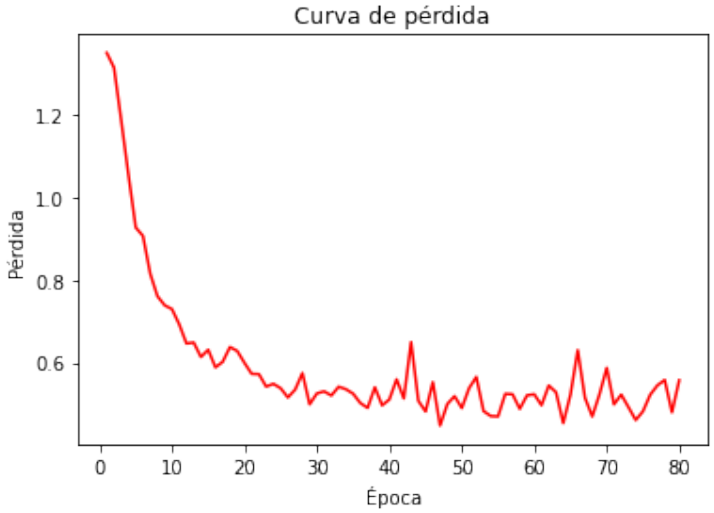

Figura 16: Pérdida en función de la época validando con el conjunto de datos de test.

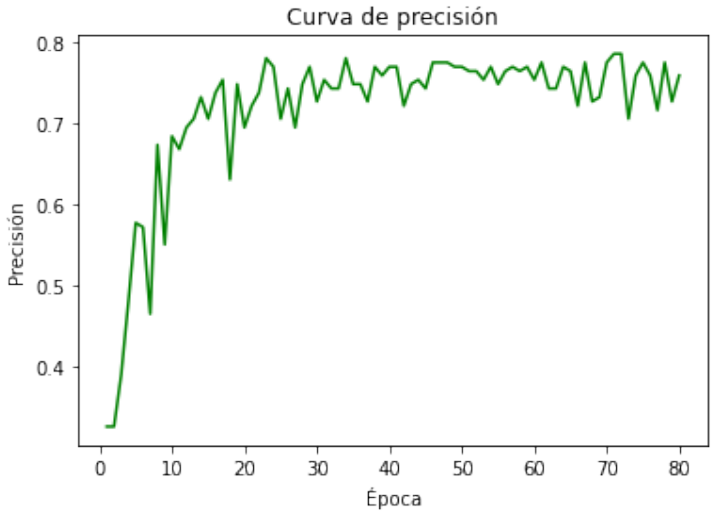

Figura 17: Precisión en función de la época validando con el conjunto de datos de test.

En efecto, los resultados van mejorando sobre un conjunto de datos que no se ha usado previamente para entrenar la red. Vemos que la pérdida tiende a decrecer de una forma aproximadamente monótona salvo pequeñas oscilaciones y converge a un cierto valor pequeño de pérdida. Dicho valor no es idénticamente cero porque el modelo tiene una capacidad limitada que quizá podría mejorarse disponiendo de un conjunto de datos mayor. Por otro lado, la preci-

sión tiende a crecer de una forma aproximadamente monótona salvo pequeñas oscilaciones y la precisión a la que converge es del mismo orden que la de modelos anteriores.

Finalmente, se muestra la representación de las curvas ROC:

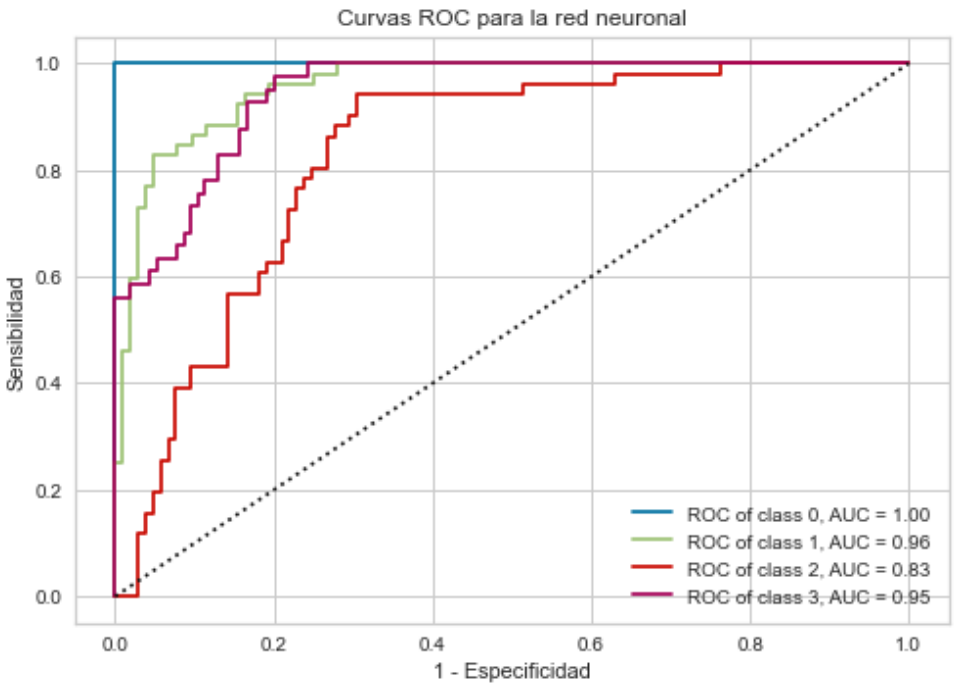

Figura 18: Curvas ROC para cada clase y área bajo cada curva (AUC).

mostrando un comportamiento equivalente al del bosque aleatorio aunque ligeramente mejor como se evidencia en el cálculo de las áreas mostrado en la leyenda.

6.2. Redes regresoras

En el caso de la regresión cuyas salidas son números reales, la función de activación de la capa de salida es directamente la identidad y sólo habrá una neurona en dicha capa. Por ello, tras la aplicación de la función identidad, pasamos directamente a la medición de la pérdida a través de otra función. Típicamente, se puede utilizar el error cuadrático medio, o directamente la mitad del error cuadrático (la única parte que importa matemáticamente en el proceso de optimización es la diferencia cuadrática) junto con un término de regularización similar al descrito antes en el caso de la clasificación. Así, dicha función de coste sería

$$J = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 + \frac{\alpha}{2}||W||_2^2 \tag{30}$$

donde $\hat{y}_i$ son los valores predichos e y_i son los valores suministrados.

Retomando la definición de (23), se obtiene en este caso $R^2 = 0,9359$, cercano a 1 y un poco más que en el caso del bosque. Comparando las bondades enteras del conjunto de test con las predichas encajadas en *bins* de números enteros, se observa un claro parecido:

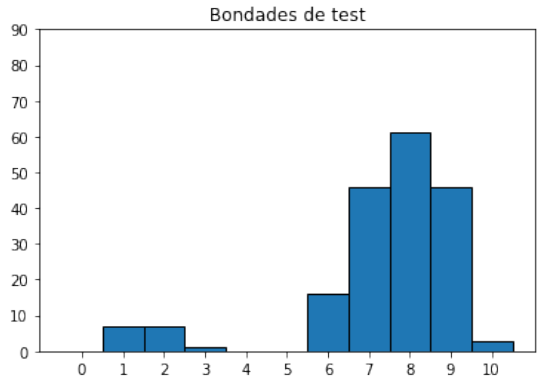

Figura 19: Bondades etiquetadas en el conjunto de test.

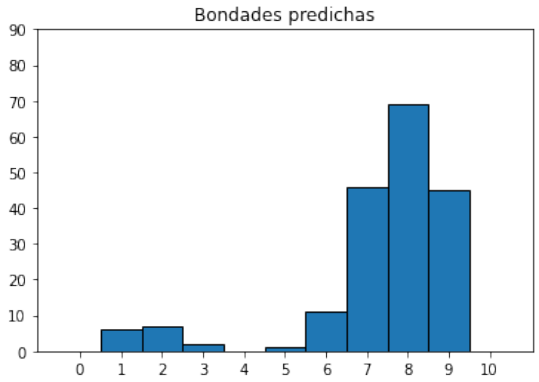

Figura 20: Predicciones de la red neuronal.

7. Conclusión

En este trabajo, se ha llevado a cabo un tratamiento completo de un conjunto de datos cedidos por la empresa Distromel S.A. acerca de las actividades de recogida de residuos urbanos que arrojan grandes cantidades de datos que deben procesarse inteligentemente. Esto ha incluido el cuidadoso análisis de los datos para conocer su estructura y las mejores formas de agruparlos y extraer información relevante a partir de ellos. El proceso de análisis ha comenzado por un rastreo de las bases de datos de la empresa para seleccionar un conjunto de datos suficientemente representativo. Tras ello, se han determinado las variables más relevantes para la determinación de la bondad con la que se ejecuta una orden de trabajo. Después, se han realizado diversos análisis estadísticos y modelos basados en algoritmos de inteligencia artificial y se han estudiado los resultados.

El análisis de los árboles de decisión ha permitido observar su eficacia a la hora de obtener una partición concreta del conjunto de datos que sea fácilmente interpretable de forma visual. Este análisis se puede refinar un poco más a través de bosques aleatorios para obtener mejores clasificadores y regresores, aunque a costa de la interpretabilidad visual. Lo mismo sucede con las redes neuronales, las cuales son capaces de aproximar funciones complejas, aunque sacrificando intuición. Para este conjunto de datos, teniendo en cuenta su forma y tamaño, se ha observado que los bosques aleatorios y las redes neuronales ofrecen resultados muy similares. Para la forma particular de construir la bondad de una orden que se ha mostrado en este trabajo, parece que las redes han ofrecido un resultado ligeramente mejor. No obstante, cuando los datos de entrada se dan en forma tabular, los árboles de decisión y los bosques aleatorios plantean un proceso de entrenamiento más simple y con menos parámetros que ajustar que las redes neuronales. La libertad de configurar más parámetros puede ser algo positivo, pero también una desventaja por el trabajo previo de preprocesamiento sin que necesariamente una red neuronal vaya a ofrecer unos resultados sustancialmente mejores en todos los problemas.

En cuanto a líneas futuras, el análisis de datos aquí presentado puede ser de utilidad para la empresa colaboradora de cara a conocer la estructura de un conjunto de datos típico y proponer modelos que podrían rehacerse en el futuro desde un punto de vista más orientado hacia el cliente. Así, este trabajo podría suponer un punto de partida hacia modelos más avanzados con un mayor

número de datos y variables con los que realizar predicciones más precisas, más objetivas y más robustas ante la presencia de datos atípicos. Por otro lado, los modelos aquí presentados también podrían utilizarse para crear otros modelos más reducidos. Por ejemplo, este sería el caso de la modelización de las actividades de limpieza viaria que también gestiona informáticamente la empresa. En ese caso, no contaríamos con las variables de contenedores recogidos, pero sí con las variables relacionadas con la geometría de las rutas. De esta forma, podrían crearse nuevos modelos haciendo una particularización de los presentados en este trabajo.

Desde el contexto del grado en Física, este trabajo ha resultado ser muy útil como aplicación de conocimientos de programación, análisis estadístico avanzado y comparación de modelos complejos. Asimismo, dada la relevancia que tienen en el mundo actual la inteligencia artificial y el tratamiento inteligente de los datos, este trabajo ha sido una muy provechosa puesta en práctica de una serie de conocimientos que, sin duda, le resultarán de utilidad a un servidor en los años venideros.

Bibliografía

[1] Amat, J. (2020). *Machine learning con Python y Scikit-learn.*
`https://www.cienciadedatos.net/documentos/py06_machine_learning_python_scikitlearn.html`

[2] Besse, P., Guillouet, B., Loubes, J.M. y Royer, F. (2015). *Review & Perspective for Distance Based Trajectory Clustering.*
`https://arxiv.org/pdf/1508.04904.pdf`

[3] Breiman, L., Friedman, J., Olshen, R. y Stone, C. (1984). *Classification and Regression Trees.*

[4] Brownlee, J. (2018). *A Gentle Introduction to the Bootstrap Method.*
`https://machinelearningmastery.com/a-gentle-introduction-to-the-bootstrap-method/`

[5] Géron, A. (2017). *Hands-On Machine Learning with Scikit-Learn & TensorFlow.*

[6] Grosse, R. (2019). *Lecture 5: MultiLayer Perceptrons.*
`https://www.cs.toronto.edu/~mren/teach/csc411_19s/lec/lec10_notes1.pdf`

[7] Hastie, T., Tibshirani, R. y Friedman, J. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (segunda edición).*
`https://web.stanford.edu/~hastie/Papers/ESLII.pdf`

[8] Kuhn, M. y Johnson, K. (2013). *Applied Predictive Modeling.*

[9] Narkhede, S. (2018). *Understanding AUC - ROC Curve.*
`https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5`

[10] *Página web de la librería scikit-learn.*
`https://scikit-learn.org`

[11] *Python TensorFlow Tutorial – Build a Neural Network* (2020).
`https://adventuresinmachinelearning.com/python-tensorflow-tutorial`